UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/990,802	11/13/2001	Eitan Farchi	SVL920010003US1/CA1259	3720

46159 7590 02/21/2008

SUGHRUE MION PLLC
USPTO CUSTOMER NO WITH IBM/SVL
2100 PENNSYLVANIA AVENUE, N.W.
WASHINGTON, DC 20037

EXAMINER	
MITCHELL, JASON D	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
02/21/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/990,802	FARCHI ET AL.	
	Examiner	Art Unit	
	Jason Mitchell	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 January 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date. _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-22 are pending in this application.

Response to Arguments

Applicant's arguments have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1, 5-8, 12-15, and 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,673,387 to Chen et al. (Chen) in view of US 6,170,083 to Abl-Tabatabai (Abl-Tabatabai).

Regarding Claim 1: Chen discloses identifying the computer program for which the persistent code coverage data should be collected (col. 2, lines 47-50 'a software system is partitioned in to basic code entities');

dividing the program source code statements of said computer program into a plurality of code coverage tasks (col. 2 lines 47-50 'partitioned into basic code entities'; col. 3, lines 46-61 'Functions are the basic entities that execute program semantics');

generating a persistent (col. 7, line 64-col. 8, line 2 'generates the C program database 177') unique name for each of the code coverage tasks of said plurality of code coverage tasks (col. 9, lines 1-3 'all entities have the attributes … kind, … name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind');

inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program (col. 7, lines 7-9 'adding instrumentation to the code, which results in instrumented C source code'; *Note that 'function trace lists 161 and 163 through 165' generated by the instrumented code contain sufficient data to determine which 'entities' or 'code coverage tasks' have been executed and consequently there must be instrumentation 'for' each 'entity'* see e.g. col. 9, lines 32-57);

compiling and linking the instrumented program into a program executable (col. 7, lines 9-11 'Instrumented C source code is then compiled by a C compiler with the results in instrumented C source code');

identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes (col. 11, lines 66-67 'determine which test units … need to be re-run');

creating a code coverage database using the code coverage tasks and the identified set of test cases (col. 9, lines 32-35 'generate an entity trace list for each test unit');

running the program executable with a test case from the identified set of test cases (col. 7, lines 30-33 'one execution for each of the N test units') and writing the

information about the test case and the coverage points that are executed into an output file (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and

processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data (col. 9, lines 32-35 'the function trace lists … are then used to generate an entity trace list').

Chen's does not disclose each of the plurality of code coverage tasks comprising a basic block of code in which the basic block of code is a set of consecutive statements with a single entry point and a single exit point.

Adl-Tabatabai teaches instrumentation wherein the code coverage tasks are basic block of code in which the basic block of code is a set of consecutive statements with a single entry point and a single exit point (col. 6, lines 11-15 "introduces instrumentation code after each basic block of code"; col. 5, lines 37-38 "A basic block of code is defined as a set of instructions in between branch instructions.")

It would have been obvious to one of ordinary skill in the art at the time the invention was made to replace Chen's instrumentation methods with those taught by Abl-Tabatabai (col. 6, lines 11-15). Those of ordinary skill in the art would have been motivated to make such a change as a known and obvious alternate implementation of

Chen's functionality (col. 7, lines 22-23 "Other well known methods for instrumenting …may also be used.")

Regarding Claims 5, 12, and 19: The rejections of claims 1, 8, and 15 are incorporated respectively; further, Chen does not disclose that the computer program comprises program source code statements written in a hardware description language. Chen does teach that his invention may be 'applied to the selective regression testing of software systems written in other languages' (col. 9, lines 22-24).

It would have been obvious to one of ordinary skill in the art at the time of the invention to implement selective regression testing, as detailed by Chen, for a hardware description language.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a system to regression test software written in languages other than C (col. 9, lines 22-24).

Regarding Claim 6: The rejection of claim 1 is incorporated; further, Chen discloses:

modifying the computer program to produce a modified version of the computer program source code (col. 10. line 50-53 'after modifications have been made');

identifying a plurality of new, modified, and deleted code coverage tasks in said modified version of the computer program source code (col. 10, lines 63-66 'entity difference list');

generating a persistent unique name (col. 9, lines 1-3 'all entities have the attributes … kind, … name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks;

inserting coverage points into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code (col. 7, lines 7-9 'adding instrumentation');

compiling and linking the instrumented modified version of the computer program source code into a modified program executable (col. 7, lines 9-11 'compiled by a C compiler');

identifying a new set of test cases from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks (col. 11, lines 66-67 'the entity difference list is used … to determine which tests units … need to be re-run');

altering the code coverage database to accommodate new, modified and deleted code coverage tasks (col. 12, lines 41-47 'new entity trace lists must be generated for each test unit') and the new set of test cases, and clearing any code coverage data for

the modified code coverage tasks (col. 12, lines 41-47 'new entity traces list must be generated') from said code coverage database;

running the modified program executable with a test case from the identified new set of test cases (col. 11, lines 66-67 're-run') and collecting code coverage data for the new and modified code coverage tasks (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and

updating the code coverage database with the collected code coverage data (col. 9, lines 32-35 'the function trace lists … are then used to generate') for the new and modified code coverage tasks;

whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved (col. 12, lines 41-47 'new entity trace lists must be generated ... covered by each of the selected test units') from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket (col. 12, lines 38-40 'selected test units are re-run in order to test the modified software system').

Regarding Claim 7: The rejection of claim 6 is incorporated; further Chen discloses changing the version indicator (col. 8, lines 59-63 'checksum is used … to determine whether an entity has been changed' and col. 11, lines 26-27 'the checksum … is retrieved from the second C program database'). While not explicitly stated, the

checksum does identify the version of the entity in that it distinguishes between two different versions of said entity.

Regarding Claim 8: Chen discloses an apparatus for collecting persistent code coverage data for a program, the persistent code coverage data being stored in a code coverage database associated with the program (col. 6, lines 50-52 'the external storage device … may be used for the storage of data), the program comprising program source code statements (col. 5, lines 33-52), the apparatus comprising:

a computer system having a data storage device (col. 6, lines 50-52 'the external storage device … may be used for the storage of data') connected thereto, wherein the data storage device stores the code coverage database, and

one or more computer programs (col. 6, lines 50-52 'the external storage device … may be used for the storage of … computer program code') executed by the computer system for:

identifying the computer program for which the persistent code coverage data should be collected (col. 2, lines 47-50 'a software system');

dividing the program source code statements of said computer program into a plurality of code coverage tasks (col. 2 lines 47-50 'partitioned into basic code entities');

generating a persistent unique name for each of the code coverage tasks (col. 9, lines 1-3 'all entities have the attributes … kind, … name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') of said plurality of code coverage tasks;

inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program (col. 7, lines 7-9 'adding instrumentation'),

compiling and linking the instrumented program into a program executable (col. 7, lines 9-11 'compiled by a C compiler'),

identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes (col. 11, lines 66-67 'determine which test units … need to be re-run');

creating a code coverage database using the code coverage tasks and the identified set of test cases (col. 9, lines 32-35 'generate an entity trace list for each test unit');

running the program executable with a test case from the identified set of test cases (col. 7, lines 30-33 'for each of the N test units') and writing the information about the test case and the coverage points that are executed into an output file (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and

processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data (col. 9, lines 32-35 'the function trace lists … are then used to generate an entity trace list').

Chen's does not disclose each of the plurality of code coverage tasks comprising a basic block of code in which the basic block of code is a set of consecutive statements with a single entry point and a single exit point.

Adl-Tabatabai teaches instrumentation wherein the code coverage tasks are basic block of code in which the basic block of code is a set of consecutive statements with a single entry point and a single exit point (col. 6, lines 11-15 "introduces instrumentation code after each basic block of code"; col. 5, lines 37-38 "A basic block of code is defined as a set of instructions in between branch instructions.")

It would have been obvious to one of ordinary skill in the art at the time the invention was made to replace Chen's instrumentation methods with those taught by Abl-Tabatabai (col. 6, lines 11-15). Those of ordinary skill in the art would have been motivated to make such a change as a known and obvious alternate implementation of Chen's functionality (col. 7, lines 22-23 "Other well known methods for instrumenting …may also be used.")

Regarding Claim 13: The rejection of claim 8 is incorporated; further, Chen discloses modifying the computer program to produce a modified version of the computer program source code (col. 10. line 50-53 'after modifications have been made');

identifying a plurality of new, modified, and deleted code coverage tasks (col. 10, lines 63-66 'entity difference list') in said modified version of the computer program source code;

generating a persistent unique name (col. 9, lines 1-3 'all entities have the attributes … kind, … name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks;

inserting coverage points (col. 7, lines 7-9 'adding instrumentation') into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code;

compiling and linking the instrumented modified version of the computer program source code (col. 7, lines 9-11 'compiled by a C compiler') into a modified program executable,

identifying a new set of test cases (col. 11, lines 66-67 'the entity difference list is used … to determine which tests units … need to be re-run') from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks;

altering the code coverage database to accommodate new, modified and deleted code coverage tasks (col. 12, lines 41-47 'new entity trace lists must be generated for each test unit') and the new set of test cases, and clearing any code coverage data for

the modified code coverage tasks (col. 12, lines 41-43 'new entity traces list must be generated') from said code coverage database;

running the modified program executable with a test case from the identified new set of test cases (col. 11, lines 66-67 're-run') and collecting code coverage data for the new and modified code coverage tasks (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and

updating the code coverage database with the collected code coverage data (col. 9, lines 32-35 'the function trace lists … are then used to generate') for the new and modified code coverage tasks;

whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved (col. 12, liens 41-47 'new entity trace lists must be generated ... covered by each of the selected test units') from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket (col. 11, lines 66-67 'which test units … need to be re-run').

Regarding Claim 14: The rejection of claim 13 is incorporated; further Chen discloses changing the version indicator (col. 8, lines 59-63 'checksum is used … to determine whether an entity has been changed' and col. 11, lines 26-27 'the checksum … is retrieved from the second C program database'). While not explicitly stated, the

checksum does identify the version of the entity in that it distinguishes between two different versions of said entity.

Regarding Claim 15: Chen discloses an article of manufacture comprising a program storage device (col. 6, lines 50-52 'the external storage device) readable by a computer and tangibly embodying one or more programs of instructions (col. 6, lines 50-52 'the external storage device … may be used for the storage of … computer program code') executable by the computer to perform method steps for collecting persistent code coverage data for a computer program (col. 1, lines 11-14 'selective regression'), the computer program comprising program source code statements (col. 5, lines 33-52), the method comprising the steps of:

identifying the computer program for which the persistent code coverage data should be collected (col. 2, lines 47-50 'a software system');

dividing the program source code statements of said computer program into a plurality of code coverage tasks (col. 2 lines 47-50 'partitioned into basic code entities');

generating a persistent unique name for each of the code coverage tasks (col. 9, lines 1-3 'all entities have the attributes … kind, … name' and col. 11, lines 20-21 'two entities match if they have the same name and entity kind') of said plurality of code coverage tasks;

inserting coverage points into the computer program source code for each of the code coverage tasks to produce an instrumented program (col. 7, lines 7-9 'adding instrumentation'),

compiling and linking the instrumented program into a program executable (col. 7, lines 9-11 'compiled by a C compiler'),

identifying a set of test cases from a plurality of test cases to be run for the code coverage data collection purposes (col. 11, lines 66-67 'determine which test units … need to be re-run');

creating a code coverage database using the code coverage tasks and the identified set of test cases (col. 9, lines 32-35 'generate an entity trace list for each test unit');

running the program executable with a test case from the identified set of test cases (col. 7, lines 30-33 'for each of the N test units') and writing the information about the test case and the coverage points that are executed into an output file (col. 7, lines 40-44 'generates a function trace list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and

processing the information contained in the output file into code coverage data and populating the code coverage database with said code coverage data (col. 9, lines 32-35 'the function trace lists … are then used to generate an entity trace list').

Chen's does not disclose each of the plurality of code coverage tasks comprising a basic block of code in which the basic block of code is a set of consecutive statements with a single entry point and a single exit point.

Adl-Tabatabai teaches instrumentation wherein the code coverage tasks are basic block of code in which the basic block of code is a set of consecutive statements with a single entry point and a single exit point (col. 6, lines 11-15 “introduces instrumentation code after each basic block of code"; col. 5, lines 37-38 “A basic block of code is defined as a set of instructions in between branch instructions.”)

It would have been obvious to one of ordinary skill in the art at the time the invention was made to replace Chen’s instrumentation methods with those taught by Abl-Tabatabai (col. 6, lines 11-15). Those of ordinary skill in the art would have been motivated to make such a change as a known and obvious alternate implementation of Chen's functionality (col. 7, lines 22-23 “Other well known methods for instrumenting …may also be used.”)

Regarding Claim 20: The rejection of claim 15 is incorporated; further, Chen discloses:

modifying the computer program to produce a modified version of the computer program source code (col. 10. line 50-53 ‘after modifications have been made’);

identifying a plurality of new, modified, and deleted code coverage tasks (col. 10, lines 63-66 ‘entity difference list’) in said modified version of the computer program source code;

generating a persistent unique name (col. 9, lines 1-3 ‘all entities have the attributes … kind, … name’ and col. 11, lines 20-21 ‘two entities match if they have the

same name and entity kind') for each of the new and modified code coverage tasks of said plurality of new, modified and deleted code coverage tasks;

inserting coverage points (col. 7, lines 7-9 'adding instrumentation') into the modified version of the computer program source code for each of the new and modified code coverage tasks to produce an instrumented modified version of the computer program source code;

compiling and linking the instrumented modified version of the computer program source code (col. 7, lines 9-11 'compiled by a C compiler') into a modified program executable,

identifying a new set of test cases (col. 11, lines 66-67 'the entity difference list is used … to determine which tests units … need to be re-run') from a plurality of test cases to be run for the code coverage data collection purposes on the new and modified code coverage tasks;

altering the code coverage database to accommodate new, modified and deleted code coverage tasks (col. 12, lines 41-47 'new entity trace lists must be generated for each test unit') and the new set of test cases, and clearing any code coverage data for the modified code coverage tasks (col. 12, lines 41-43 'new entity traces list must be generated') from said code coverage database;

running the modified program executable with a test case from the identified new set of test cases (col. 11, lines 66-67 're-run') and collecting code coverage data for the new and modified code coverage tasks (col. 7, lines 40-44 'generates a function trace

list'), until all the test cases have been run (col. 7, line 32 'for each of the N test units'); and

updating the code coverage database with the collected code coverage data (col. 9, lines 32-35 'the function trace lists … are then used to generate') for the new and modified code coverage tasks;

whereby the previously collected code coverage data for the non-affected code coverage tasks is preserved (col. 12, liens 41-47 'new entity trace lists must be generated ... covered by each of the selected test units') from a previous version of the computer program to the modified version of said computer program eliminating the need for running the entire test bucket (col. 11, lines 66-67 'which test units … need to be re-run').

Regarding Claim 21: The rejection of claim 20 is incorporated; further Chen discloses changing the version indicator (col. 8, lines 59-63 'checksum is used … to determine whether an entity has been changed' and col. 11, lines 26-27 'the checksum … is retrieved from the second C program database'). While not explicitly stated, the checksum does identify the version of the entity in that it distinguishes between two different versions of said entity.

Claims 2-3, 9-10, 16-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,673,387 to Chen et al. (Chen) in view of US 6,170,083 to Abl-

Tabatabai (Abl-Tabatabai) in view of 'Managing data through naming standards' by Winder, Software, IEEE, Volume: 7, Issue: 4, July 1990 (Winder).

Regarding Claims 2, 9, and 16: The rejections of claim 1, 8 and 15 are incorporated respectively; further Chen does not disclose using naming conventions. But does disclose attributes of each entity provide a unique identifier for said entity (col. 11, lines 20-21 'two entities match if they have the same name and entity kind.').

Winder teaches using a naming convention (pg. 85, col. 1, par. 3 'A naming standard fights … ambiguity') in an analogous art for the purpose of managing data and eliminating ambiguity (pg. 85, col. 1, par. 4 'eliminating the ambiguity').
It would have been obvious to a person of ordinary skill in the art at the time of the invention to include a naming convention as taught by Winder in the naming of coverage tasks ('entities') as disclosed in Chen.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide names for entities that would enable one to access the entities (Winder pg. 84, col. 3, par. 1-2 'determines your ability to access the information'), in Chen's disclosed system.

Regarding Claim 3, 10 and 17: the rejection of claims 2, 9 and 16 is incorporated; further, Chen does not disclose the naming convention comprising a module name, a

version and a unique task identifier But does disclose maintaining similar attributes (i.e. col. 9, lines 1-3 'kind, file, name and checksum') where checksum is used to determine a function version (col. 8, lines 59-62 'determine whether an entity has been changed') Winder teaches the use of a three part naming convention (pg. 85, col. 2, par. 1 'these data-element names are called primary, class, and modifier').

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use Winder's three part naming convention (pg. 85, col. 2, par. 1) populated with the data gathered in Chen col. (col. 9, lines 1-3) to label the entities disclosed in Chen (col. 8, lines 1-4) thereby creating a unique naming convention comprising a computer program module name (col. 8, lines 44-45 'file'), a version indicator (col. 8, lines 51-52 'checksum'), and a unique code coverage task identifier (col. 8, lines 46-47 'name').

The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide names for entities that would enable one to access the entities (Winder pg. 84, col. 3, par. 1-2 'determines your ability to access the information'), in Chen's disclosed system.

Claims 4, 11, and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,673,387 to Chen et al. (Chen) in view of US 6,170,083 to Abl-Tabatabai (Abl-Tabatabai) in view of USPN 5,778,169 to Reinhardt (Reinhardt).

Regarding Claims 4, 11, and 18: The rejections of claims 1, 8, and 15 are incorporated, respectively; further, Chen does not disclose that the code coverage database comprises a table, the table comprising a row for each test case in said identified set of test cases and a column for each code coverage task of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task. Chen does foresee the need to allow a user to determine which test units would need to be re-run if a hypothetical change were made to the software system (col. 12, lines 55-58).

Reinhardt teaches the code coverage database comprises a table (col. 6, lines 16-18 'test coverage matrix'), the table comprising a row for each test case (col. 6, lines 24-25 'names of the tests') in said identified set of test cases and a column for each code coverage task (col. 6, lines 25-26 'coverage point names') of said plurality of code coverage tasks, said column comprising an indicator at each row indicating coverage status for said code coverage task (col. 6, lines 26-27 'the relationship between the tests and coverage points'), in an analogous art for the purpose of providing programmers with knowledge of which coverage points are executed by which tests (col. 6, lines 34-35 'view the test coverage matrix')

It would have been obvious to a person of ordinary skill in the art at the time of the invention to use the matrix taught by Reinhardt in the regression test system of Chen to display the program's code coverage data to a programmer (Chen col. 12, lines 55-58). The modification would have been obvious because one of ordinary skill in the art would have been motivated to allow programmers to easily identify regression tests that test possible source code changes (Reinhardt, col. 2, lines 62-63).

Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over USPN 5,673,387 to Chen et al. (Chen) in view of US 6,170,083 to Abl-Tabatabai (Abl-Tabatabai) in view of US 5,673,387 to Pastilha et al. (Pastilha).

Regarding Claim 22: The rejection of claim 1 is incorporated; further Chen does not disclose the persistent unique name comprises a string having a plurality of components reflecting the code coverage task, wherein only a component of the string of the code coverage task which has been modified is altered while maintaining non-modified components of the string.

Pastilha teaches a persistent unique name comprises a string having a plurality of components (col. 15, lines 52-54 "creates a file entitled "oan/test/filea.00001"), wherein only a component of the string of the code coverage task which has been modified is altered while maintaining non-modified components of the string (col. 15, lines 58-61 "creates a more recent (generational) file entitled "OAN/TEST/FILEA.00002").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use persistent unique names similar to those taught by Pastilha (col. 15, lines 52-54) while generating Chen's entity names (col. 9, lines 1-3 'all entities have the attributes … kind, … name'). Those of ordinary skill in the art would have been motivated to make such a change to clearly indicate the 'generation' of the entity (Chen col. 11, lines 2-4 "An entity is … changed if the source code … has changed"; Pastilha col. 15, lines 58-61 "creates a more recent (generational) file").

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Mitchell whose telephone number is (571) 272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Bullock Lewis can be reached on (571) 272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jason Mitchell/
Jason Mitchell
2/14/08

/Lewis A. Bullock, Jr./
Lewis A. Bullock, Jr.
Supervisory Patent Examiner
Art Unit 2193